Leveraging the Power of ACLs

Session 617

These are confidential sessions—please refrain from filming or taking pictures.

Enterprise IT

Leveraging the Power of ACLs

Rusty Tucker
Engineering Manager, Core Server

http://developer.apple.com/wwdc2005/

Introduction

- Provide in depth information about Access Control Lists (ACLs) in Mac OS X Tiger Server
  - Overview of access control in Tiger Server
  - File System ACL internals
  - Tiger ACLs—a developer's point of view
  - Best practices for managing your network with ACLs
  - Q&A with the ACL experts

Tiger Server Access Control Options

Rusty Tucker
Engineering Manager, Core Server

http://developer.apple.com/wwdc2005/

Types of Access Control in Tiger Server

- POSIX Permissions
- File System Access Control Lists (ACLs)
- Service Access Control Lists (SACLs)

Service Access Control Lists

http://developer.apple.com/wwdc2005/

Service Access Control Lists

- Control login access to services

Service Access Control Lists

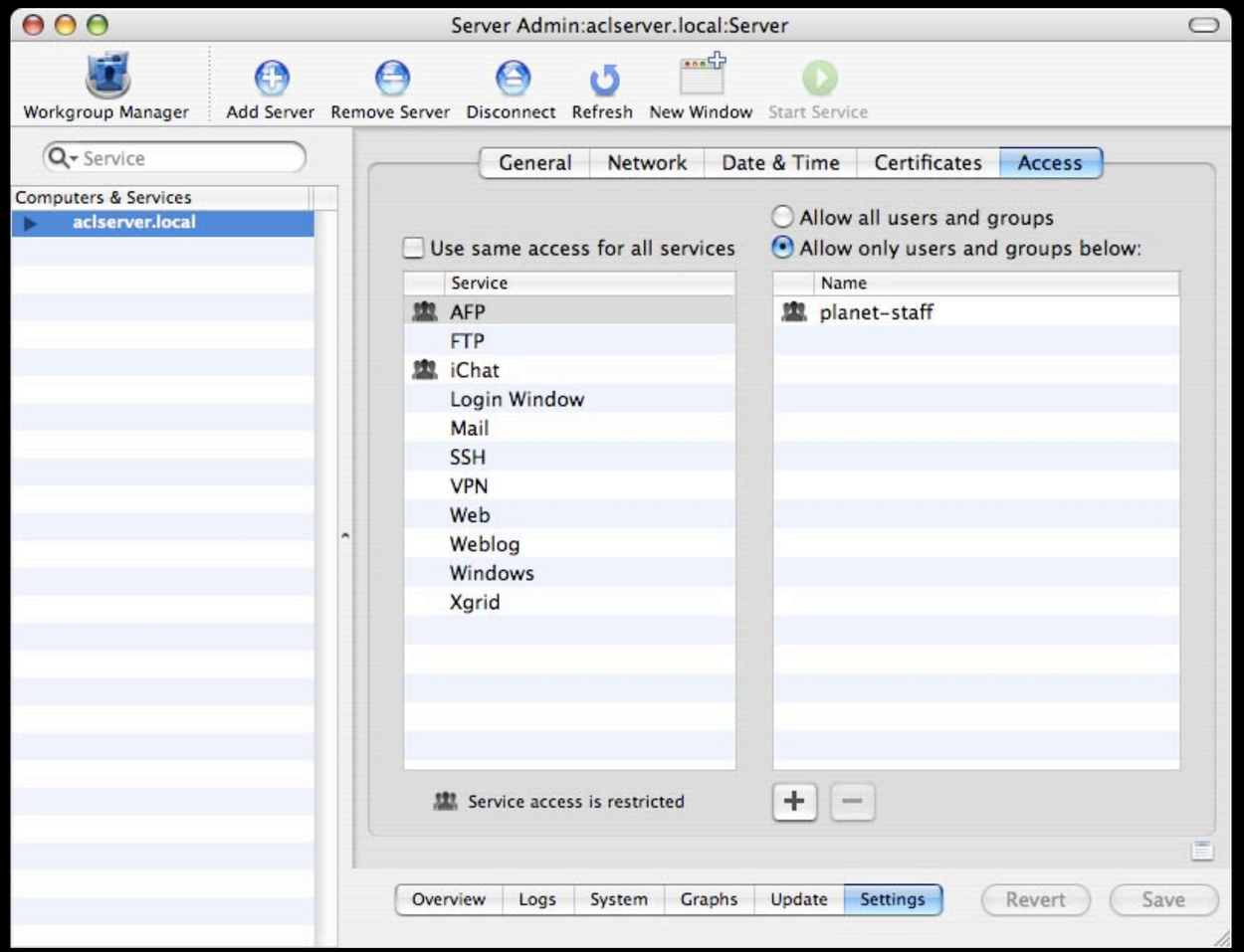
- Control login access to services
- SACL support is prevalent in Tiger Server

Service Access Control Lists

- Control login access to services
- SACL support is prevalent in Tiger Server
  - Apple File Service (AFP)
  - File Transfer Protocol (FTP)
  - iChat
  - Login Window
  - SSH
  - VPN
  - Web
  - Weblog
  - Windows Services
  - XGrid

Service Access Control Lists

Server Admin Interface



Service Access Control Lists

Service ACL Membership Checking

```
#include <membership.h >

int CanAccessService( uid_t userID, gid_t wellKnownGroupID )
{
int isMember = 0;
int err;
uuid_t userUUID;
uuid_t groupUUID;

// get the UUID of the authenticated user
err = mbr_uid_to_uuid( userID, userUUID );
assert( err == 0 );

// get the UUID of a well known group
err = mbr_gid_to_uuid( wellKnownGroupID, groupUUID );

if( err == 0 )
// use that UUID to check membership in your services allow group
err = mbr_check_membership( userUUID , groupUUID, &isMember );
```

Service Access Control Lists

Service ACL Membership Checking

```
#include <membership.h >

int CanAccessService( uid_t userID, gid_t wellKnownGroupID )
{
int isMember = 0;
int err;
uuid_t userUUID;
uuid_t groupUUID;

// get the UUID of the authenticated user
err = mbr_uid_to_uuid( userID, userUUID );
assert( err == 0 );

// get the UUID of a well known group
err = mbr_gid_to_uuid( wellKnownGroupID, groupUUID );

if( err == 0 )
// use that UUID to check membership in your services allow group
err = mbr_check_membership( userUUID , groupUUID, &isMember );
```

Service Access Control Lists

Service ACL Membership Checking

```
#include <membership.h >

int CanAccessService( uid_t userID, gid_t wellKnownGroupID )
{
int isMember = 0;
int err;
uuid_t userUUID;
uuid_t groupUUID;

// get the UUID of the authenticated user
err = mbr_uid_to_uuid( userID, userUUID );
assert( err == 0 );

// get the UUID of a well known group
err = mbr_gid_to_uuid( wellKnownGroupID, groupUUID );

if( err == 0 )
// use that UUID to check membership in your services allow group
err = mbr_check_membership( userUUID , groupUUID, &isMember );
```

Service Access Control Lists

Service ACL Membership Checking

```
#include <membership.h >

int CanAccessService( uid_t userID, gid_t wellKnownGroupID )
{
int isMember = 0;
int err;
uuid_t userUUID;
uuid_t groupUUID;

// get the UUID of the authenticated user
err = mbr_uid_to_uuid( userID, userUUID );
assert( err == 0 );

// get the UUID of a well known group
err = mbr_gid_to_uuid( wellKnownGroupID, groupUUID );

if( err == 0 )
// use that UUID to check membership in your services allow group
err = mbr_check_membership( userUUID , groupUUID, &isMember );
```

Service Access Control Lists

Service ACL Membership Checking

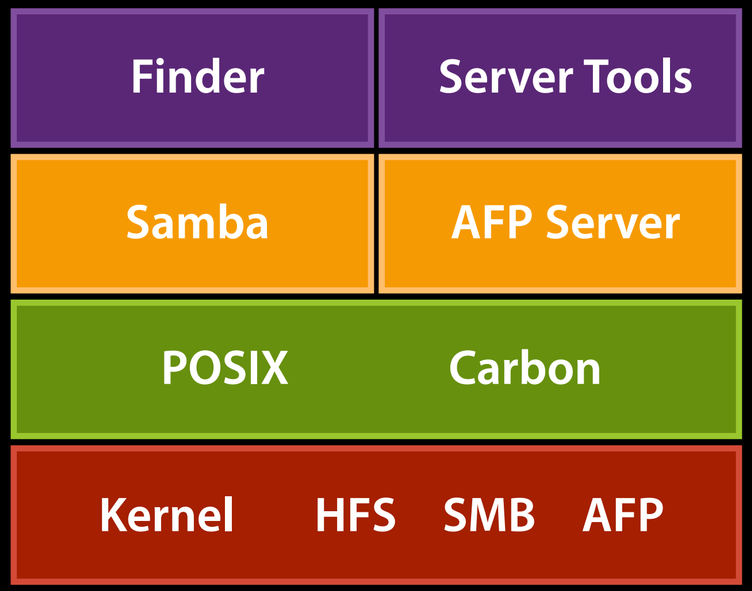
```
if( err == 0 )
// use that UUID to check membership in your services allow group
err = mbr_check_membership( userUUID , groupUUID, &isMember );

// check for errors and return authentication error when user not
permitted
if ((err && ENOENT != err) || (!err && !isMember))
return EAUTH;
else
return 0;
}
```

POSIX Permissions

- Still there—recommended for many uses
- Provide control of access to files and folders based on three categories
  - Owner
  - Group
  - Everyone
- Three basic permissions bits for each of the categories
  - Read
  - Write
  - Execute/Traverse

File System Access Control Lists (ACLs)

- Provides an extension to the POSIX permissions
  - Not an exclusive replacement
- Greater flexibility in permission management
  - Allow/Deny
  - Inheritance
  - Multiple users and groups
  - Finer granularity in permissions



File System Access Control Lists (ACLs)

- “Server Only” feature for Tiger
- Foundation exists in client
- Positioned for use by system admins
  - Security
  - Workflow
  - Interoperatibilty

File System Access Control Lists (ACLs)

Tool Support for ACLs

File System Access Control Lists (ACLs)

Tool Support for ACLs

- Workgroup Manager
  - Enable ACL support on HFS volumes
  - Create and edit Groups, Users

File System Access Control Lists (ACLs)

Tool Support for ACLs

- Workgroup Manager
  - Enable ACL support on HFS volumes
  - Create and edit Groups, Users
- Finder
  - ACL in “Get Info”
  - Copy support

File System Access Control Lists (ACLs)

Tool Support for ACLs

- Command line
  - `fsaclctl`
    - Enable ACL support on HFS volumes

File System Access Control Lists (ACLs)

Tool Support for ACLs

- Command line
  - `fsaclctl`
    - Enable ACL support on HFS volumes
  - `dseditgroup`
    - Manipulate group records

File System Access Control Lists (ACLs)

Tool Support for ACLs

- Command line
  - `fsaclctl`
    - Enable ACL support on HFS volumes
  - `dseditgroup`
    - Manipulate group records
  - `ls`
    - Displays ACLs using -e option

File System Access Control Lists (ACLs)

Tool Support for ACLs

- Command line
  - `fsaclctl`
    - Enable ACL support on HFS volumes
  - `dseditgroup`
    - Manipulate group records
  - `ls`
    - Displays ACLs using -e option
  - `chmod`
    - Edit ACLs

File System Access Control Lists (ACLs)

Tool Support for ACLs

- Command line
  - `fsaclctl`
    - Enable ACL support on HFS volumes
  - `dseditgroup`
    - Manipulate group records
  - `ls`
    - Displays ACLs using -e option
  - `chmod`
    - Edit ACLs
  - `cp`
  - `gzip, gnutar/tar, cpio, pax, bzip2, scp, install`
    - Also copies Resource forks and Extended Attributes

File System ACL Internals

Mike Smith
Software Engineer, Core OS

http://developer.apple.com/wwdc2005/

File System ACL Internals

- How an ACL is composed
- How privilege is calculated
- How the inheritance model works

How an ACL Is Composed

ACL Terminology

- ACE—Access Control Entry
- ACL—Access Control List
- UID/GID—User/Group ID
- UUID—Universally Unique ID
- SID—NT Security ID

Mode	POSIX mode bits
Access Control List	List of ACEs
Owner	UID
Group	GID

How an ACL Is Composed

Extended Security Descriptor

- Functionally is a superset of ACL and POSIX mode, group and owner.
- Accessed and set by API extensions
  - chmodx/statx
  - acl_* family

How an ACL Is Composed

What is in an ACE?

- Permissions
- Allowed or Denied
- Group or user identity
- Inheritance rules

How Privilege Is Calculated

Ordering and priority with extended security

- ACL when present evaluated first
  - Allows cumulative
  - Deny results in automatic rejection
- Canonical ordering
- POSIX permissions consulted when necessary
- Some edge cases result

How Privilege Is Calculated

Making sense of the differences

	POSIX - Owner	POSIX Group or Other	ACEs Read	Write	Read & Write	Full Control
Adminstration						
Changing Permissions	✓	Never				✓
Take Ownership	Never	Never				✓
Read	✓	✓				
Read Attributes	*	*	✓	✓	✓	✓
Read Extended Attribute	*	*	✓		✓	✓
List Folder Content / Read Data	*	*	✓		✓	✓
Traverse Folder / Execute File	X Bit	X Bit	✓		✓	✓
Read Permissions	*	*	✓	✓	✓	✓
Write	✓	✓				
Write Attributes	*	*		✓	✓	✓
Write Extended Attributes	*	*		✓	✓	✓
Create Files / Write Data	*	*		✓	✓	✓
Create Folder / Append Data	*	*		✓	✓	✓
Delete	*	*		✓	✓	✓
Delete Subfolders and Files	*	*		✓	✓	✓

How Privilege Is Calculated

Making sense of the differences

	POSIX - Owner	POSIX Group or Other	ACEs Read	Write	Read & Write	Full Control
Adminstration						
Changing Permissions	✓	Never				✓
Take Ownership	Never	Never				✓
Read	✓	✓				
Read Attributes	*	*	✓	✓		
Read Extended Attribute	*	*	✓			
List Folder Content / Read Data	*	*	✓		✓	✓
Traverse Folder / Execute File	X Bit	X Bit	✓		✓	✓
Read Permissions	*	*	✓	✓	✓	✓
Write	✓	✓				
Write Attributes	*	*		✓	✓	✓
Write Extended Attributes	*	*		✓	✓	✓
Create Files / Write Data	*	*		✓	✓	✓
Create Folder / Append Data	*	*		✓	✓	✓
Delete	*	*		✓	✓	✓
Delete Subfolders and Files	*	*		✓	✓	✓

Only way to explicitly give “Take Ownership” is with ACL

How Privilege Is Calculated

Making sense of the differences

	POSIX - Owner	POSIX Group or Other	ACEs Read	Write	Read & Write	Full Control
Adminstration						
Changing Permissions	✓	Never				✓
Take Ownership	Never	Never				✓
Read	✓	✓				
Read Attributes	*	*	✓	✓	✓	✓
Read Extended Attribute	*	*	✓		✓	✓
List Folder Content / Read Data	*	*	✓		✓	✓
Traverse Folder / Execute File	X Bit	X Bit	✓		✓	✓
Read Permissions	*	*	✓	✓	✓	✓
Write	✓	✓				
Write Attributes	*	*		✓		✓
Write Extended Attributes	*	*		✓		✓
Create Files / Write Data	*	*		✓		
Create Folder / Append Data	*	*		✓		
Delete	*	*		✓	✓	✓
Delete Subfolders and Files	*	*		✓	✓	✓

With ACLs, the user can delete the folder itself, regardless of the parent setting

How the Inheritance Model Works

Inheritance controlled by bits in the ACE

- Defines set of ACEs applied to new object
  - Not all of the ACL need be inherited
- ACE bits that control the policy
  - Apply to child folders
  - Apply to child files
  - Apply to all descendants
  - Apply to this folder

How the Inheritance Model Works

Rules implemented by Kernel and User tools

- Inherit on create
- No change on move
- As applied by Copy Engines
  - Finder
  - cp

How Tiger ACLS Fit in Mixed Networks

Rusty Tucker
Engineering Manager, Core Server

http://developer.apple.com/wwdc2005/

How Tiger ACLs Fit in Mixed Networks

A few flavors of fitting in

- Tiger Server with
  - Windows Clients
  - Panther AFP Clients
  - Tiger AFP Clients
- TigerSMBClient in a Windows network

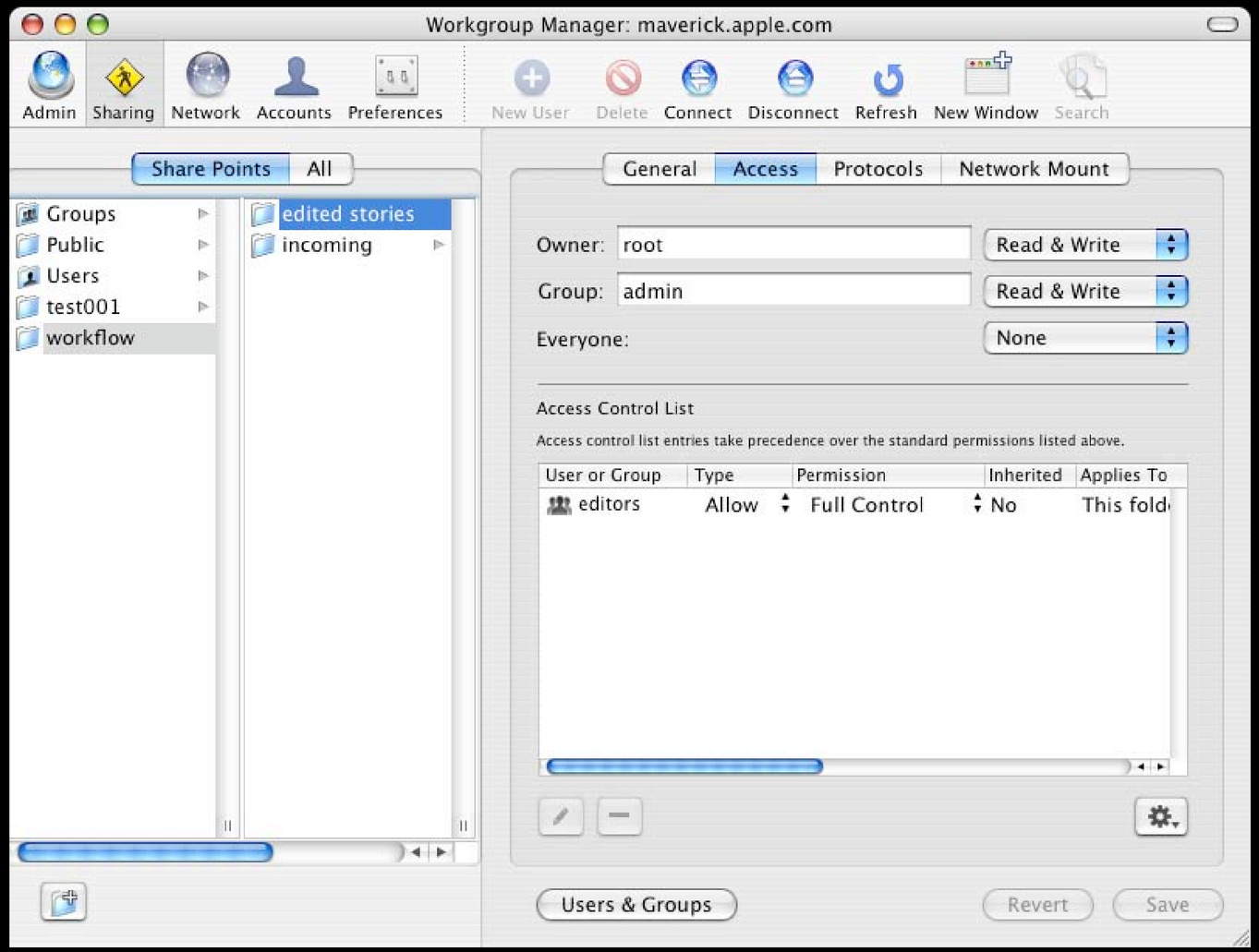
How Tiger ACLs Fit in Mixed Networks

Windows Clients and Tiger Servers

- Key goal is interoperatibility
- Mac OS X Tiger clients fully respect Windows file permissions
- Tiger ACLs+ Samba mods make Tiger look like a Win2003 Server

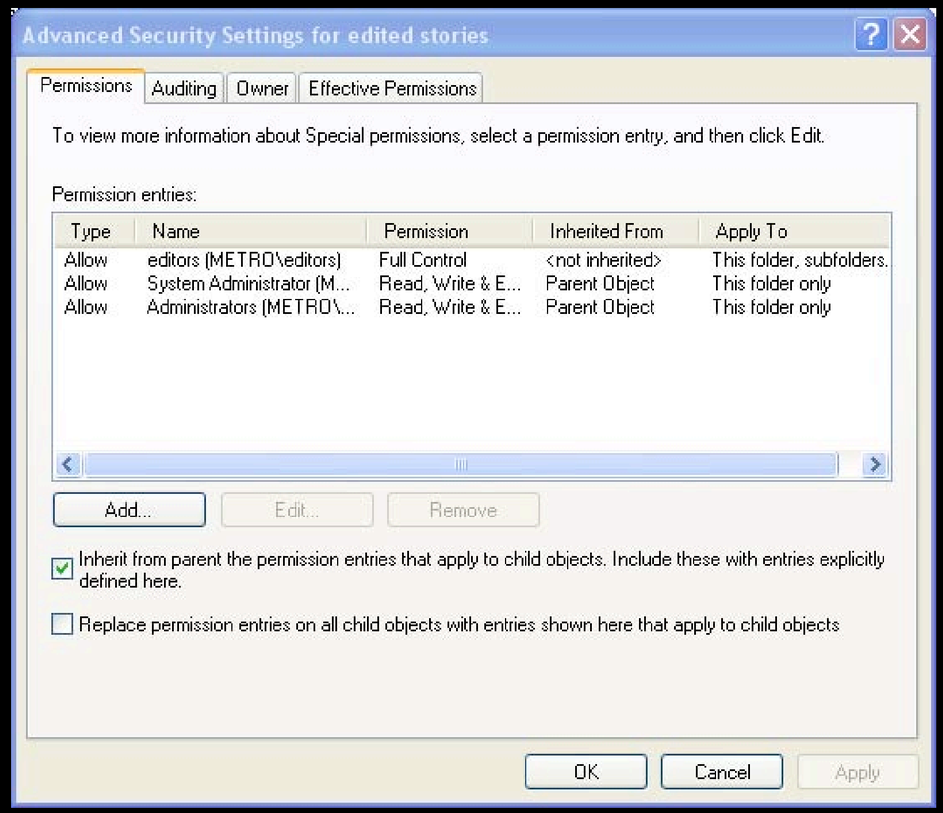
Tiger Server and Windows Clients

High level of compatibility



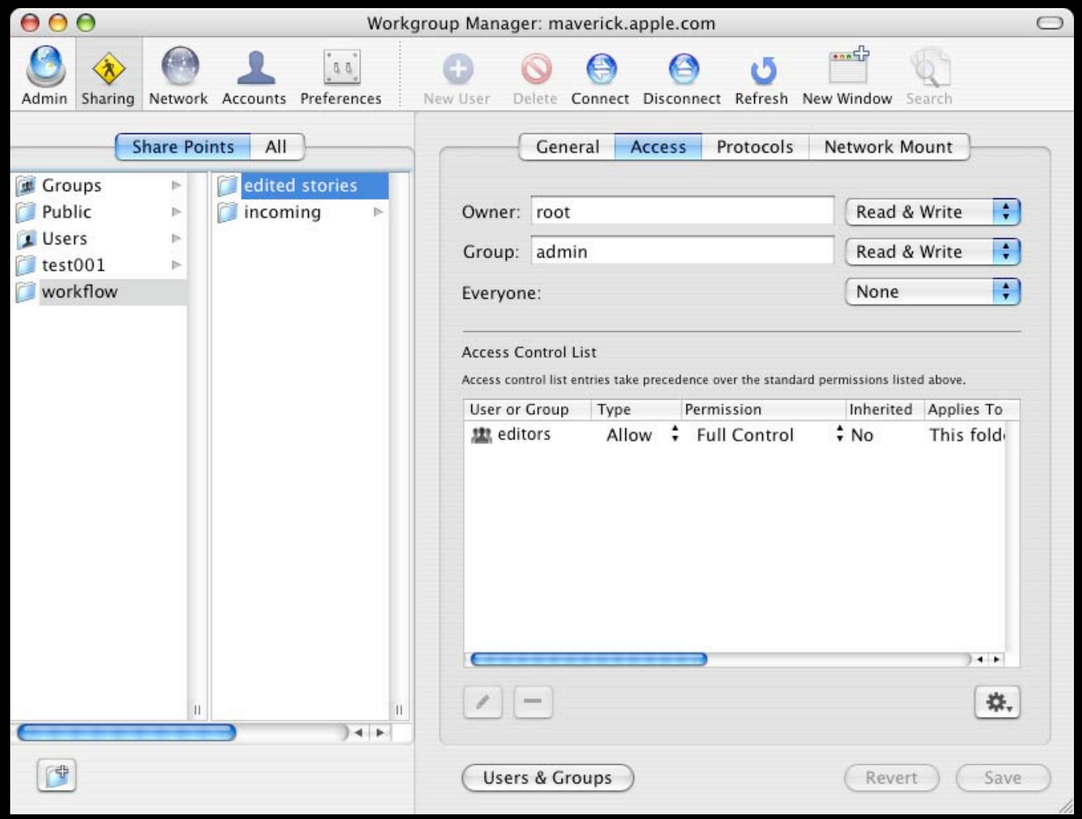
Tiger Server and Windows Clients

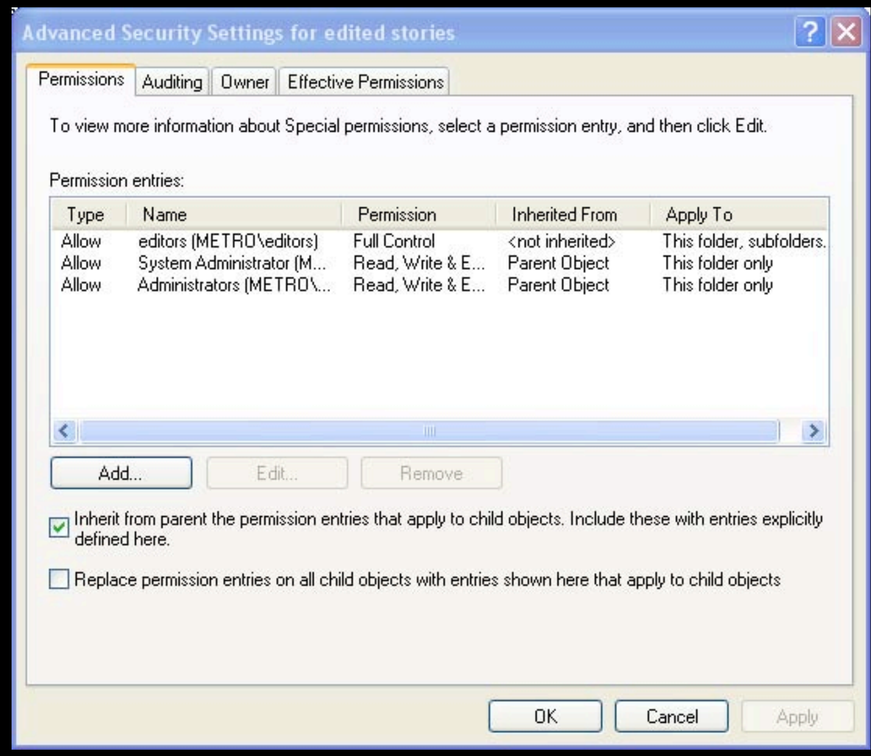
High level of compatibility



Tiger Server and Windows Clients

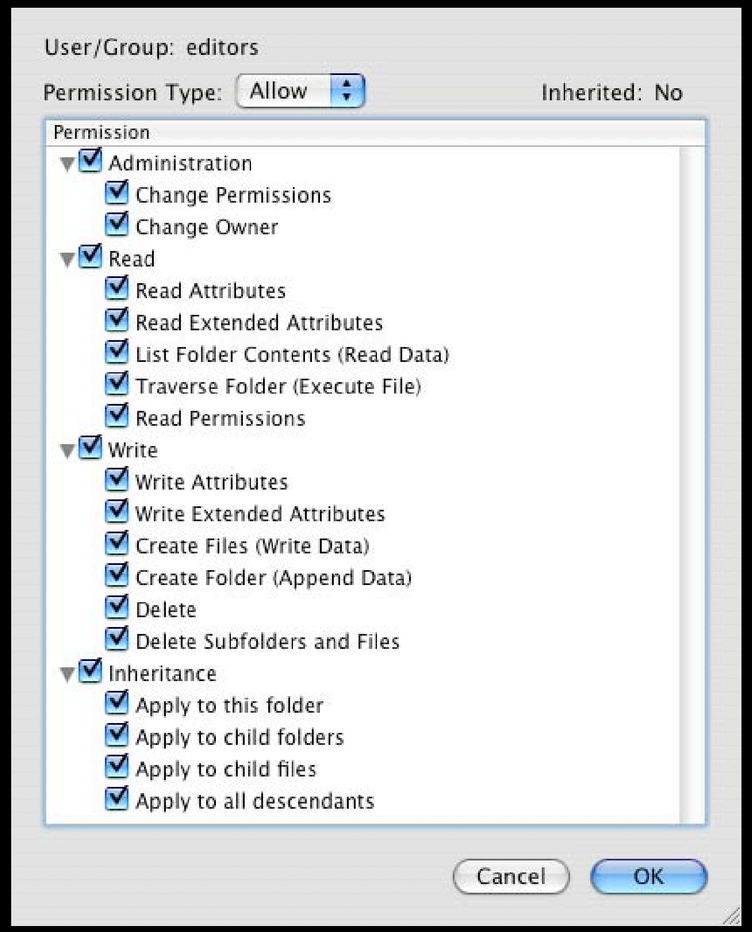
High level of compatibility

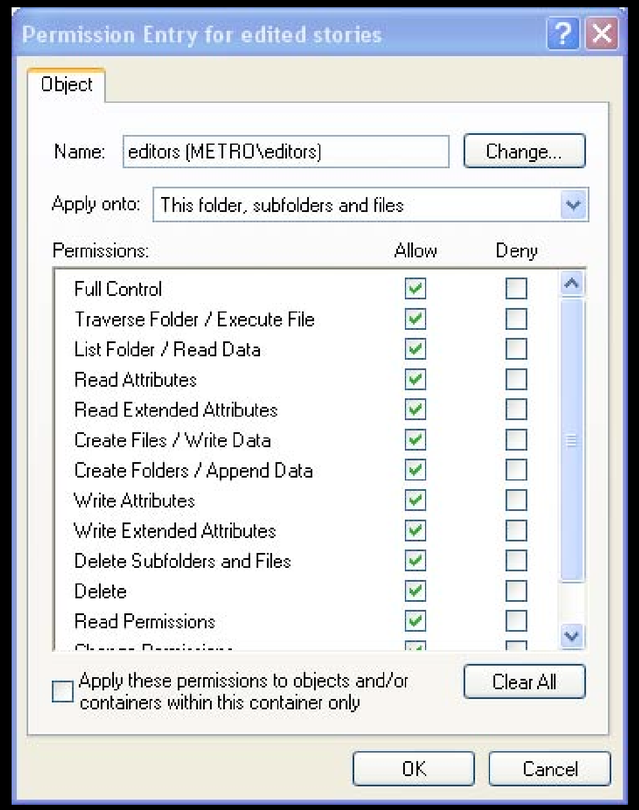




Tiger Server and Windows Clients

High level of compatibility





How Tiger ACLs Fit in Mixed Networks

Windows Clients with Tiger Server

- Matching permissions and inheritance
- POSIX mode is foreign
  - Expressed as ‘inherited’ ACE on read
  - Partial mirroring from ACEs on write
- Maintains highest degree of interop + functionality for mixed network

Tiger ACLs

A Developer's Point of View

Kevin Hoyt
Software Engineer, Tivoli Storage Manager, IBM

http://developer.apple.com/wwdc2005/

IBM Tivoli Storage Manager Overview

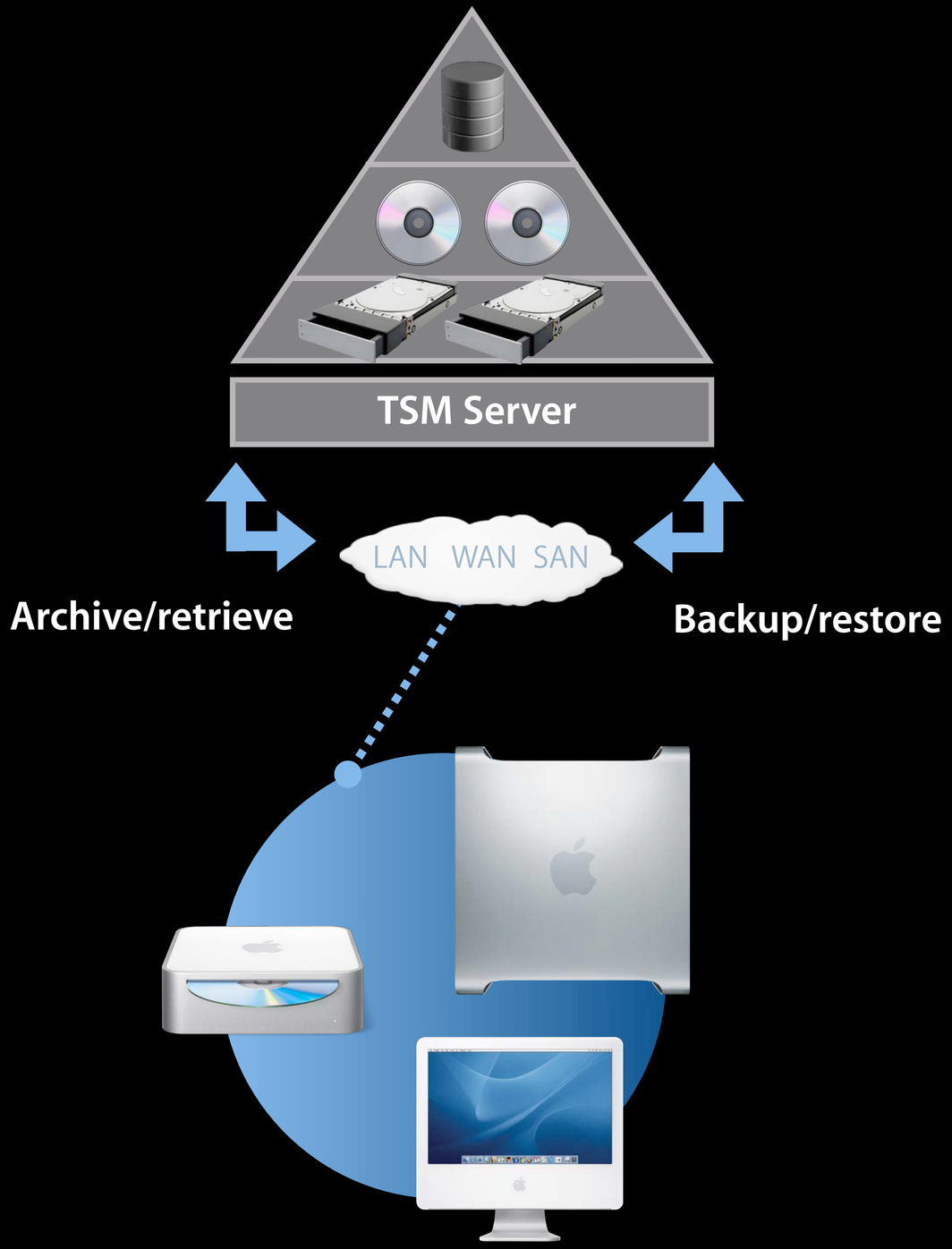
http://developer.apple.com/wwdc2005/

© 2005 IBM Corporation

Disclaimer

- This presentation describes future plans and strategies for IBM Tivoli Storage Management software products
- IBM reserves the right to change our product plans according to the Company's business needs and requirements
- These are not commitments to deliver the described features within a given time period
- We reserve the right to change product features and time periods to accommodate IBM's business objectives

© 2005 IBM Corporation



IBM Tivoli Storage Manager—TSM

- TSM is a comprehensive policy-based automated solution for centralized data protection via
  - Backup and restore
  - Archive and retrieval
  - Policy based storage management
- Customer value includes
  - Application availability through protection of data
  - Simplifies storage administration through automated policy-based management and self-configuring/optimizing technology
  - Improves the return of investment (ROI) through efficient use of storage media

IBM software group, Tivoli Storage Manager

© 2005 IBM Corporation

We Protect Your Data

Backup and Restore

- Fault tolerant automated backup
- Fault tolerant restores
- Recovery of individual files from operational errors such as… "oops" I deleted the wrong file, or I "messed up" my spreadsheet!
- Point-in-time restore to get a system back to the state it was before the "bigger oops" occurred
- Automated location of files during restore such that NO manual tracking of media is required!

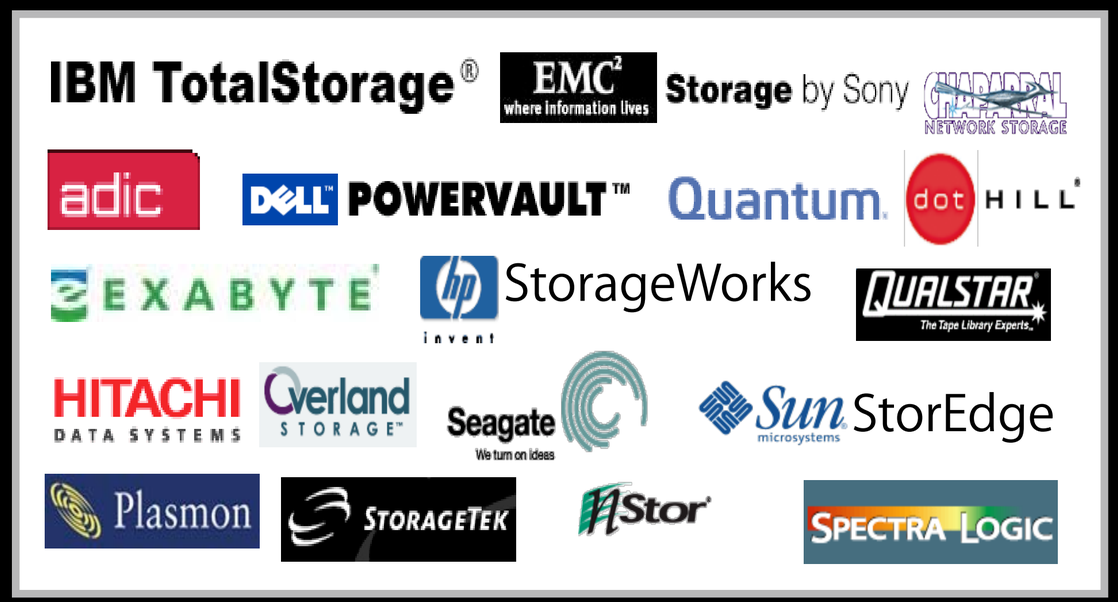
IBM®

IBM software group, Tivoli Storage Manager

© 2005 IBM Corporation

TSM Support from A to Z

- TSM Clients support from Apple and AIX to Linux, Windows, and z/OS
- TSM Servers support hundreds of storage devices including disk, tape, optical and DVD



IBM®

IBM software group, Tivoli Storage Manager

© 2005 IBM Corporation

Tivoli Storage Manager History

IBM has a 12 year history as THE enterprise backup solution for Apple!

1993	1997	1999	2002	2003
System 6 System 7	Mac OS 8	Mac OS 9	Mac OS X 10.1 Mac OS X10.2	Mac OS X 10.3 full Unicode support

IBM software group, Tivoli Storage Manager

© 2005 IBM Corporation

Tivoli Storage Manager History

IBM has a 12 year history as THE enterprise backup solution for Apple!

2004

Added the command line interface
Added the background scheduler
Added support for Case Sensitive HFS+
Added support for the Xsan filesystem

2005

Mac OS X 10.4
Adding ACL Support

IBM software group, Tivoli Storage Manager

© 2005 IBM Corporation

The Importance of ACLs

- From the System Administrator view, ACLs are a very valuable tool for controlling access to the data on their systems

- As a back up solution provider IBM recognizes the importance of managing ACLs to maintain system security

- We have experience managing ACL data on other UNIX platforms; They just happen to be new in Tiger

- The rest of this presentation will focus on what I've learned as I've added support for ACLs to Tivoli Storage Manager

IBM software group, Tivoli Storage Manager

© 2005 IBM Corporation

ACL Management Tools Available on Tiger

- To enable or disable ACLs for a file system
  - Workgroup Manager can be used on Mac OS X Server
  - fsaclctl can also be used. This tool is available for Mac OS X Server and User installs. See the Server man page, the man page is missing on the User install
  - NOTE: No matter how ACLs are enabled or disabled, the file system must be un-mounted and re-mounted!
- To set ACLs for a folder
  - Workgroup Manager can be used on Mac OS X Server
  - The chmod command can be used on Mac OS X Server and User installs
- To set ACLs for a file
  - The chmod command can be used on Mac OS X Server and User installs

IBM® IBM software group, Tivoli Storage Manager

© 2005 IBM Corporation

POSIX Functions Used for ACLs

- The prototypes are in /usr/include/sys/acl.h

```
extern acl_t acl_get_file(const char *path_p, acl_type_t type);
extern int acl_set_file(const char *path_p, acl_type_t type, acl_t acl);
extern ssize_t acl_copy_ext(void *buf_p, acl_t acl, ssize_t size);
extern acl_t acl_copy_int(const void *buf_p);
extern acl_t acl_from_text(const char *buf_p);
extern ssize_t acl_size(acl_t acl);
extern char *acl_to_text(acl_t acl, ssize_t *len_p);
```

- There are no man pages yet for:

```
acl_copy_ext()
acl_copy_int()
acl_size()
```

- If all you need is ACL support, this set of functions should be sufficient
- However, ACLs are only one aspect of file security on Tiger…

IBM software group, Tivoli Storage Manager

© 2005 IBM Corporation

The New File Security Model

- The new File Security model includes:
  - The owner ID
    - This is the uid_t we'd usually get with stat()
  - The owner Group ID
    - This is the gid_t we'd usually get with stat()
  - The owner UUID
    - The Universal Unique IDentifier of the owner
  - The Mode
    - The mode_t we'd usually get with stat()
  - The ACLs
  - The group UUID
    - The Universal Unique IDentifier of the group
- These properties are defined in /usr/include/fcntl.h in the enum filesec_property_t

© 2005 IBM Corporation

The New File Security Model

- This new model has it's own set of functions
- Some of the prototypes are in /usr/include/fcntl.h

```
filesec_t filesec_init(void);
filesec_t filesec_dup(filesec_t);
void  filesec_free(filesec_t);
int filesec_get_property(filesec_t, filesec_property_t, void *);
int filesec_set_property(filesec_t, filesec_property_t, const void *);
int filesec_unset_property(filesec_t, filesec_property_t);
int filesec_query_property(filesec_t, filesec_property_t, int *);
```

- Others are in /usr/include/stat.h

```
int chmodx_np(const char *, filesec_t);
int fchmodx_np(int, filesec_t);
int fstatx_np(int, struct stat *, filesec_t);
int lstatx_np(const char *, struct stat *, filesec_t);
int mkdirx_np(const char *, filesec_t);
int mkfifox_np(const char *, filesec_t);
int statx_np(const char *, struct stat *, filesec_t);
int umaskx_np(filesec_t);
```

- Unfortunately there are no man pages yet for any of these functions!

© 2005 IBM Corporation

The New File Security Model

- There is support for the File Security model in the Carbon API too
  - A ‘FSFileSecurityRef’ can be returned by FSGetCatalogInfo if the FSCatInfoFSFileSecurityRef bit is set in the FSCatalogInfoBitmap
  - NOTE: A FSFileSecurityRef retrieved via FSGetCatalogInfo is a copy and must be released using CFRelease() when no longer needed
  - See Files.h for more information

- You may also need some help from Core Foundation with the UUIDs
  - See CFUUID.h for more information

© 2005 IBM Corporation

How To Use filesec_t

- To retrieve the file security properties

```
/* use either statx_np() or FSGetCatalogInfo() to fill in fsec */
    struct stat myStat;
    filesec_t fsec;
    uid_t theOwner;
    gid_t theGroup;
    CFUUIDRef ownerUUID;
    CFUUIDRef groupUUID;
    mode_t theMode;
    acl_t theACL;

    statx_np(”myFile”, &myStat, fsec);
    filesec_get_property(fsec, FILESEC_OWNER, &theOwner);
    filesec_get_property(fsec, FILESEC_GROUP, &theGroup);
    filesec_get_property(fsec, FILESEC_UUID, &ownerUUID);
    filesec_get_property(fsec, FILESEC_MODE, &theMode);
    filesec_get_property(fsec, FILESEC_ACL, theACL);
    filesec_get_property(fsec, FILESEC_GRPUUID, & groupUUID);
```


© 2005 IBM Corporation

How To Use filesec_t

- To set the file security properties
  - Use `filesec_set_property()` to set the individual properites

```
filesec_t fsec;
uid_t theOwner;
gid_t theGroup;
CFUUIDRef ownerUUID;
CFUUIDRef groupUUID;
mode_t theMode;
acl_t theACL;
```

```
filesec_set_property(fsec, FILESEC_OWNER, &theOwner);
filesec_set_property(fsec, FILESEC_GROUP, &theGroup);
filesec_set_property(fsec, FILESEC_UUID, &ownerUUID);
filesec_set_property(fsec, FILESEC_MODE, &theMode);
filesec_set_property(fsec, FILESEC_ACL, theACL);
filesec_set_property(fsec, FILESEC_GRPUUID, &groupUUID);

chmodx_np(”myFile”, fsec);
```

IBM software group, Tivoli Storage Manager

© 2005 IBM Corporation

Lessons Learned

- The biggest challenge to adding support for the File Security model is the lack of documentation for the functions

- Once I understood the File Security model, I realized that the ACL functions were not enough; I needed to use the File Security functions. It was just not practical to try to manage each of the File Security properties individually

- I also realized that the File Security model is a very good solution to a common problem; how to keep all of the file security information up to date!

IBM software group, Tivoli Storage Manager

© 2005 IBM Corporation

Summary

- I do want to thank the Apple engineers that helped get me the documentation for the File Security functions. This allowed me to implement a back-up solution that includes all of the new File Security information

- Knowing that Tivoli Storage Manager will automatically manage all aspects of file security is a great benefit for the customers of Apple and IBM

- Finally, for those companies that are already using Tivoli Storage Manager we will begin beta testing our next release in July. Anyone interested can give me their business card at the end of the session

IBM software group, Tivoli Storage Manager

© 2005 IBM Corporation

ACL Best Practices

Rusty Tucker
Engineering Manager, Core Server

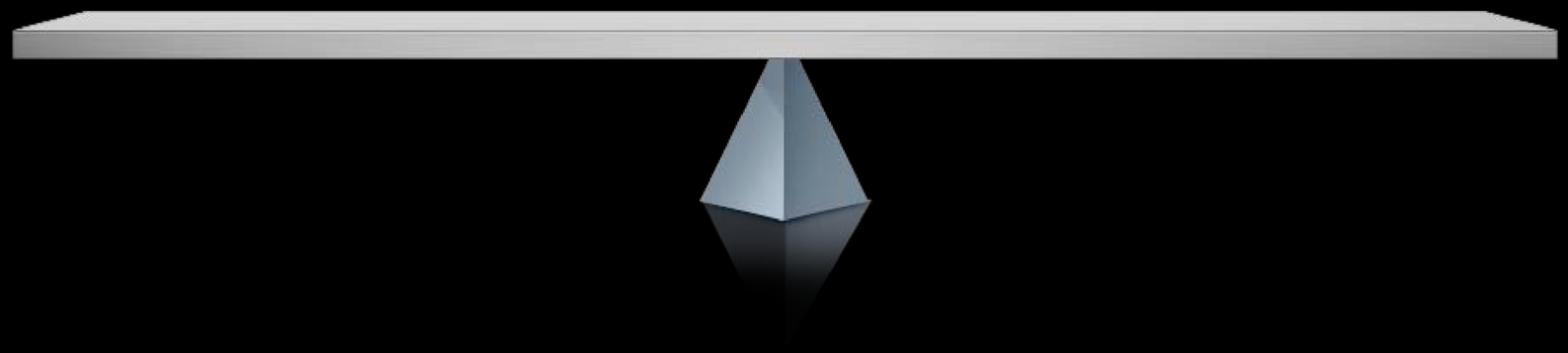
http://developer.apple.com/wwdc2005/

ACL Best Practices

Each folder can have…

Roughly 500
Possible POSIX
Combinations

Roughly 100,000
Possible ACL Permission
Combinations



ACL Best Practices

Increase server security

- Set Everyone permissions to None
- Use SACLs to limit exposure
  - Start with ‘admin’ only SSH access

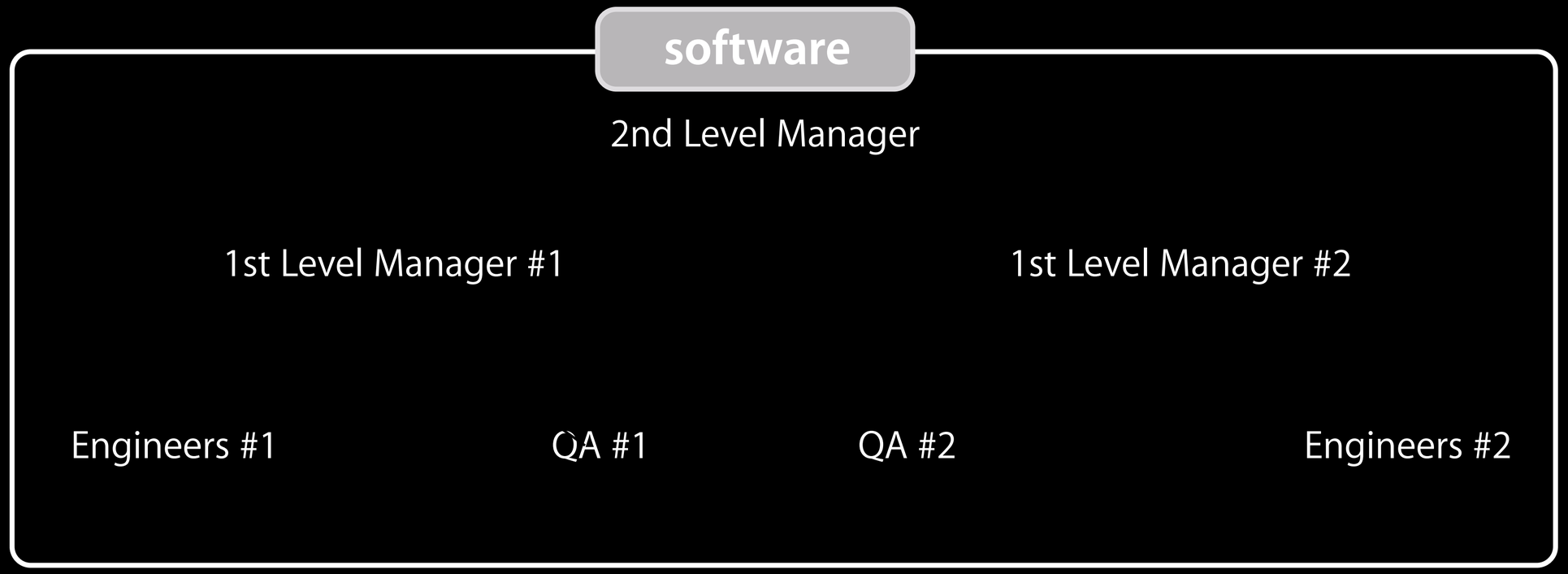
ACL Best Practices

Manage permissions at the Group level

- Assign individual permissions only as the exception
- Factor Groups when needed

ACL Best Practices

Group Factoring



ACL Best Practices

Group Factoring

Engineers #1

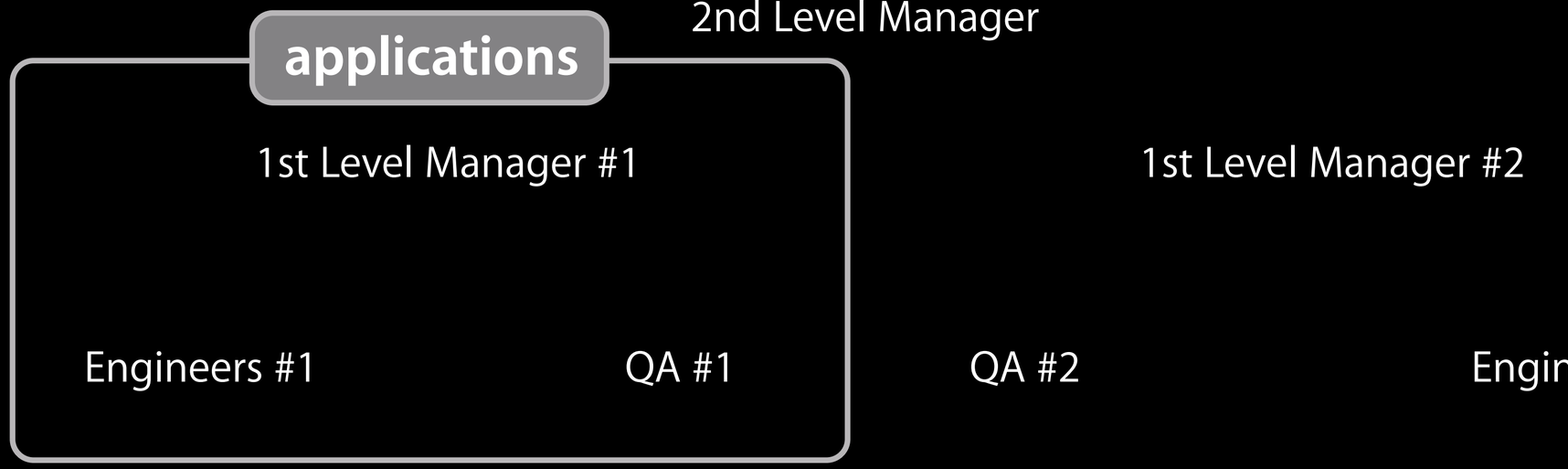
QA #1

QA #2

Engineers #2

ACL Best Practices

Group Factoring

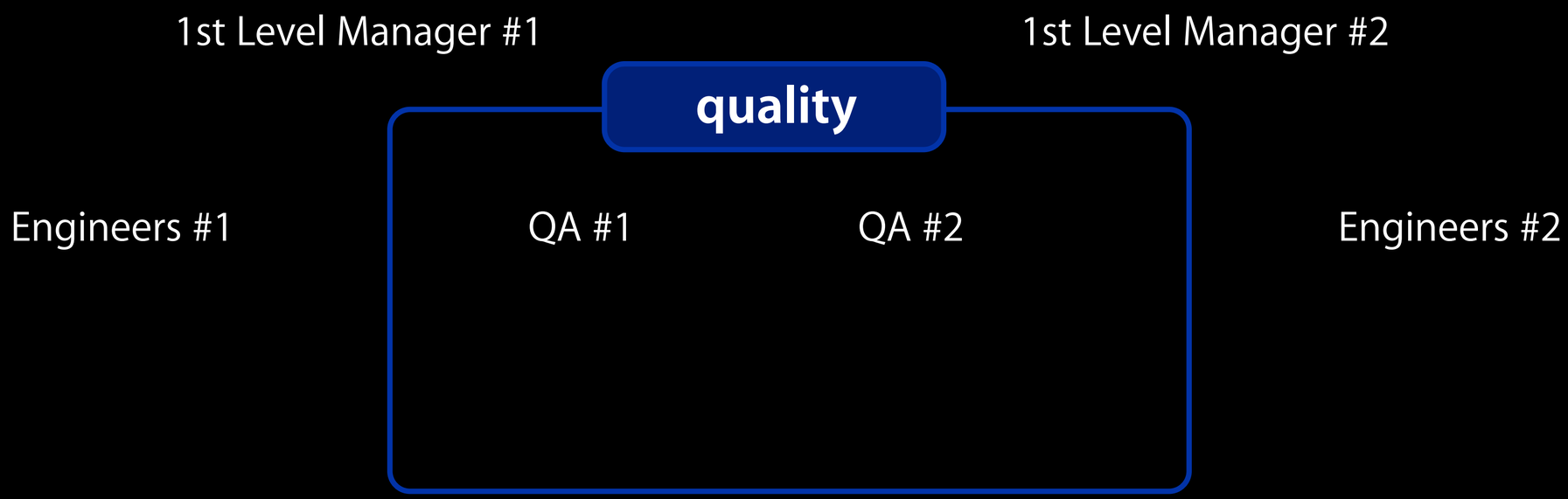


ACL Best Practices

Group Factoring

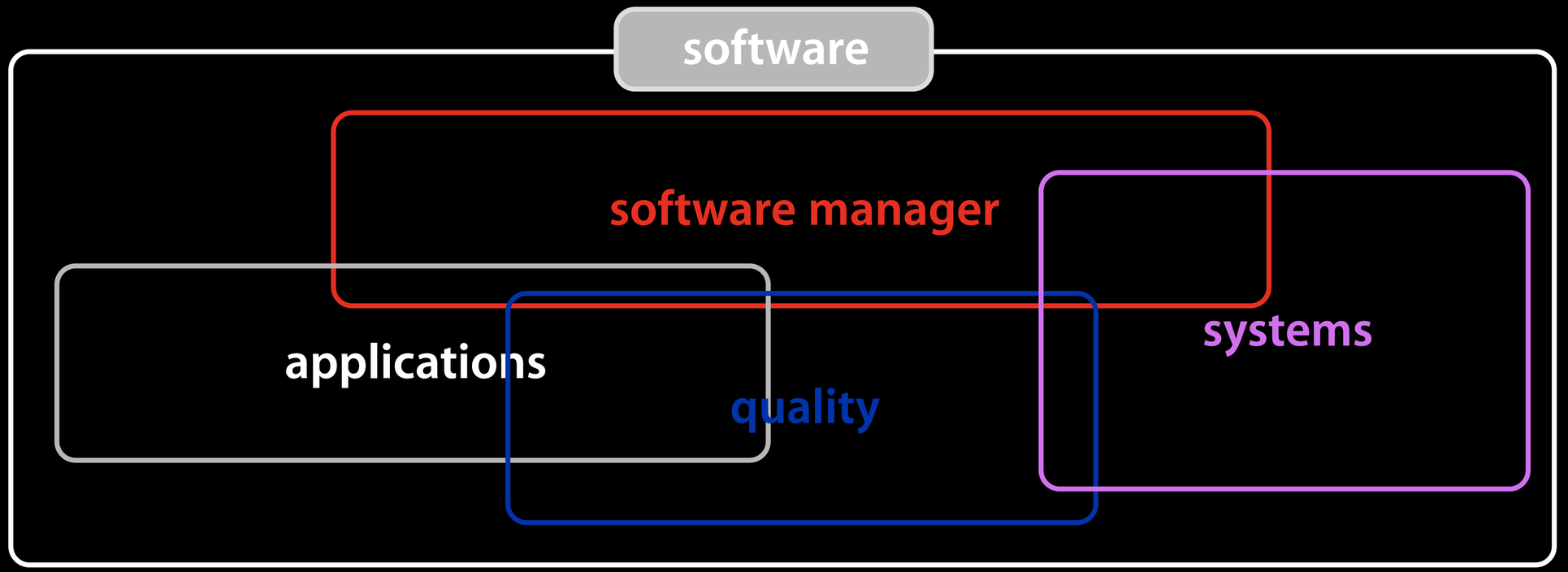
ACL Best Practices

Group Factoring



ACL Best Practices

Group Factoring



ACL Best Practices

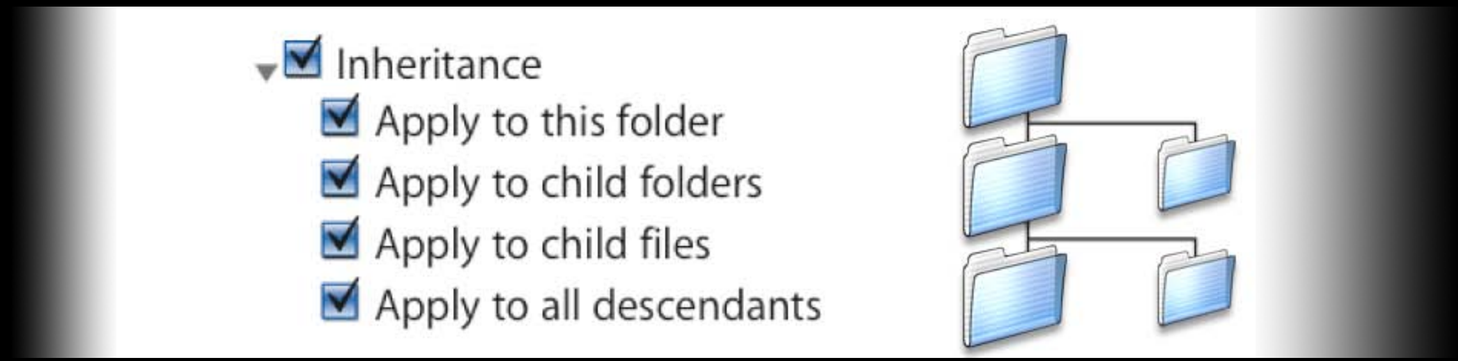
ACE Inheritance

- Use Inheritance to control permissions on files

ACL Best Practices

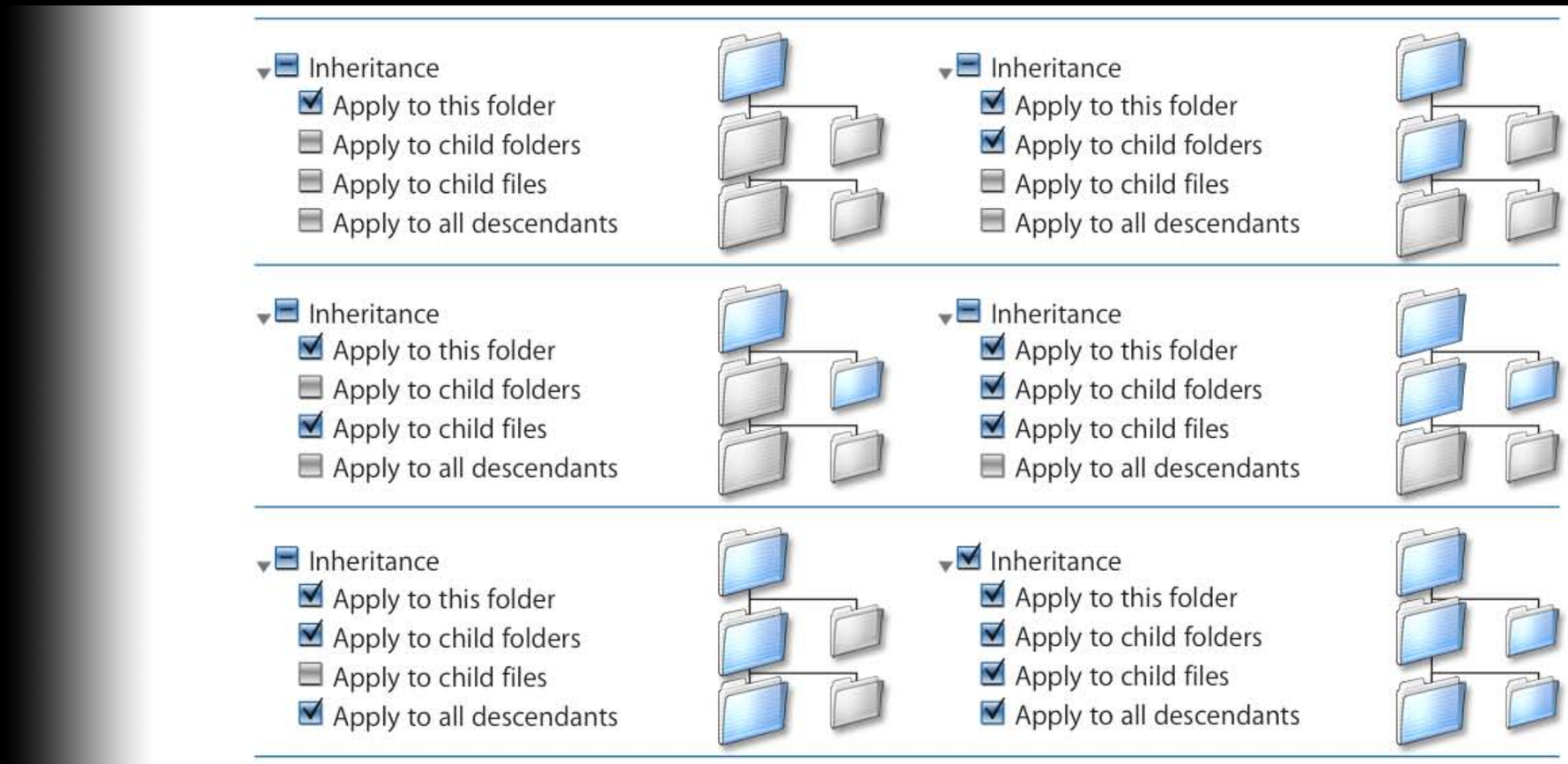
ACE Inheritance

- Use Inheritance to control permissions on files



ACL Best Practices

ACE Inheritance



ACL Best Practices

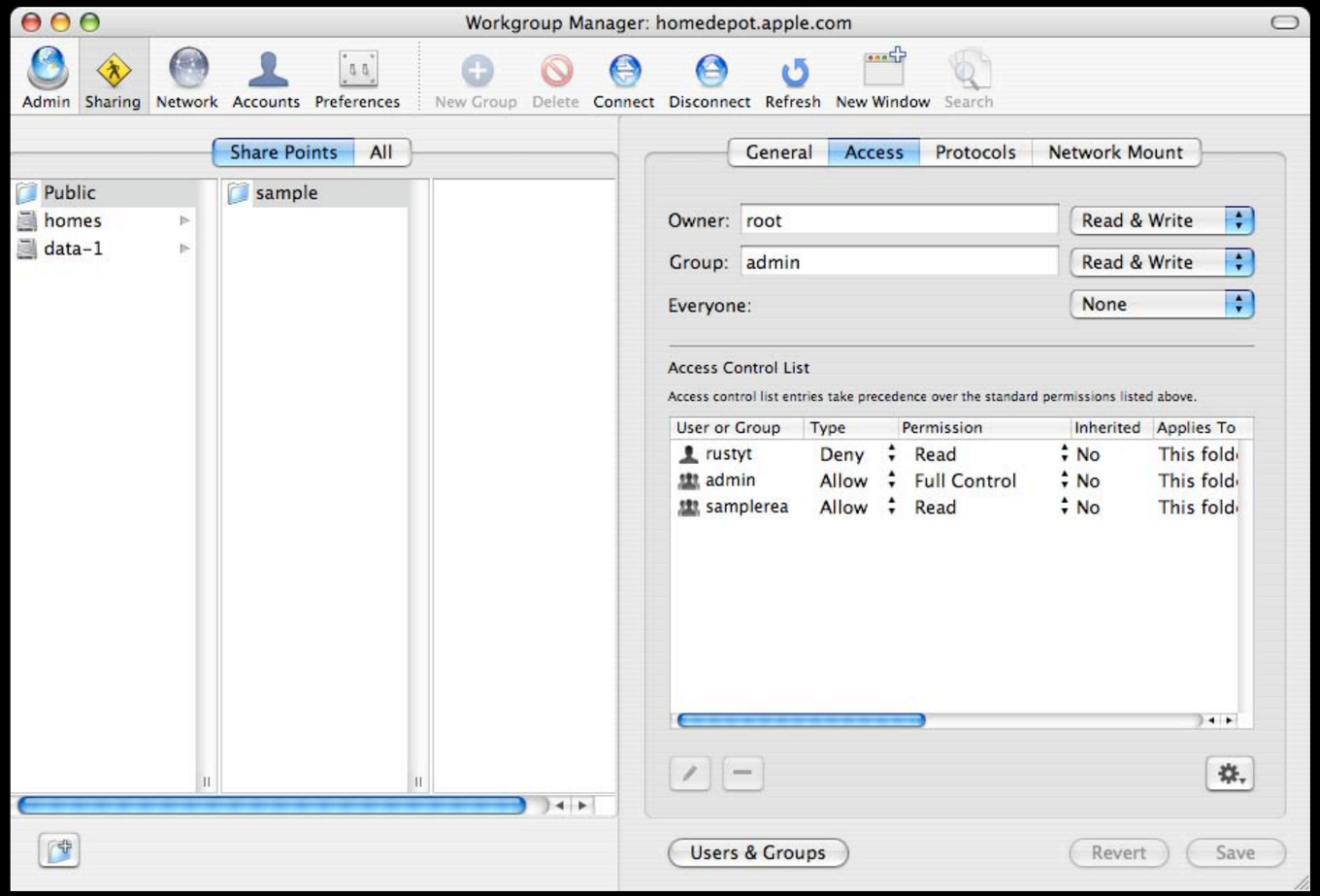
Use Propagate Permissions in the right situations

- Propagation forces inheritance, and does not have an undo
  - Remove the ACEs you added and propagate again
  - POSIX permissions can be ‘copied’ down
- Inheritance is cumulative in a hierarchy

ACL Best Practices

Use "Deny" only when you have to

- Applicable Deny ACE terminates processing



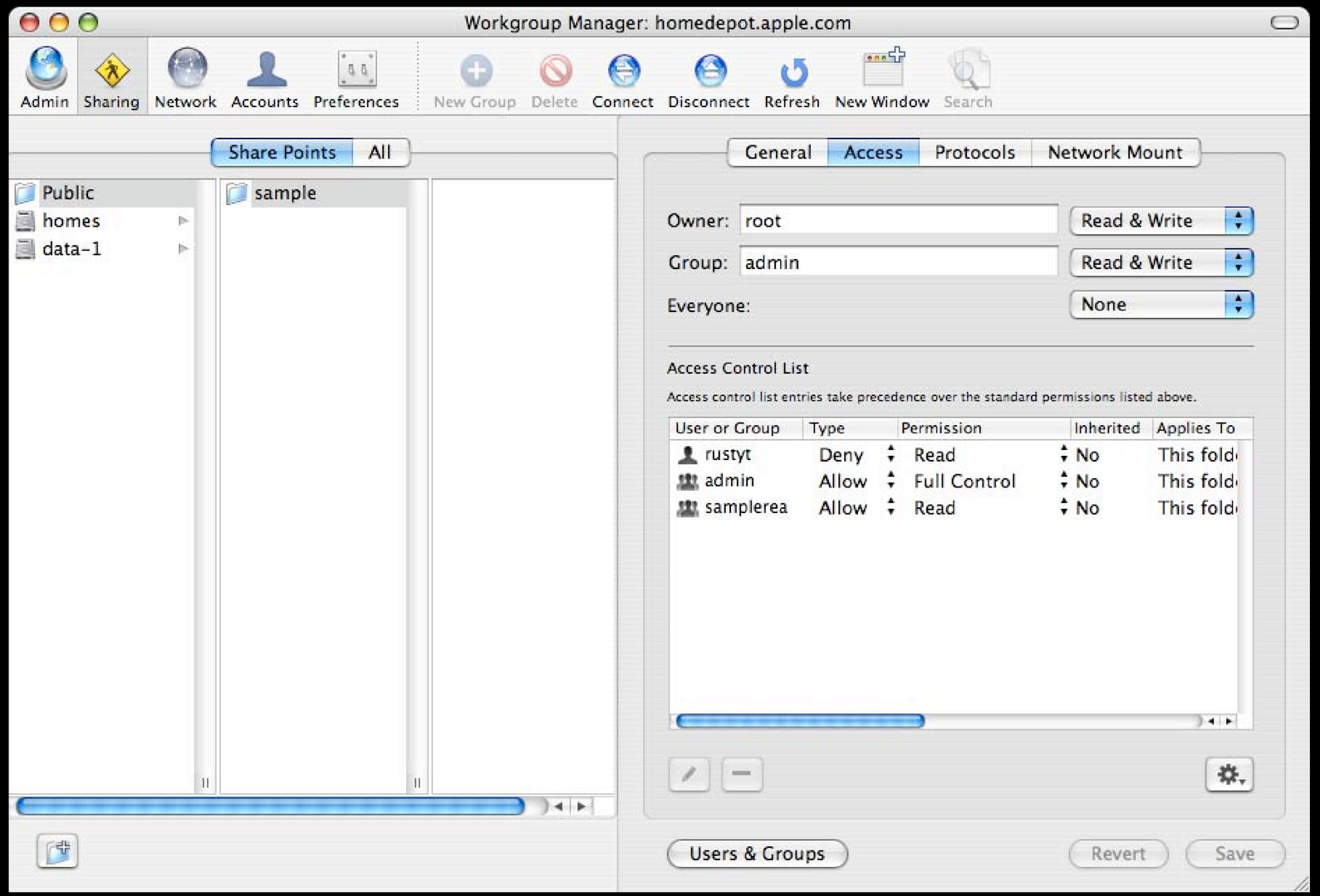
ACL Best Practices

Use the Effective Permissions Inspector

- Quickly computes a users' access rights
- Use it after changing the ACLs

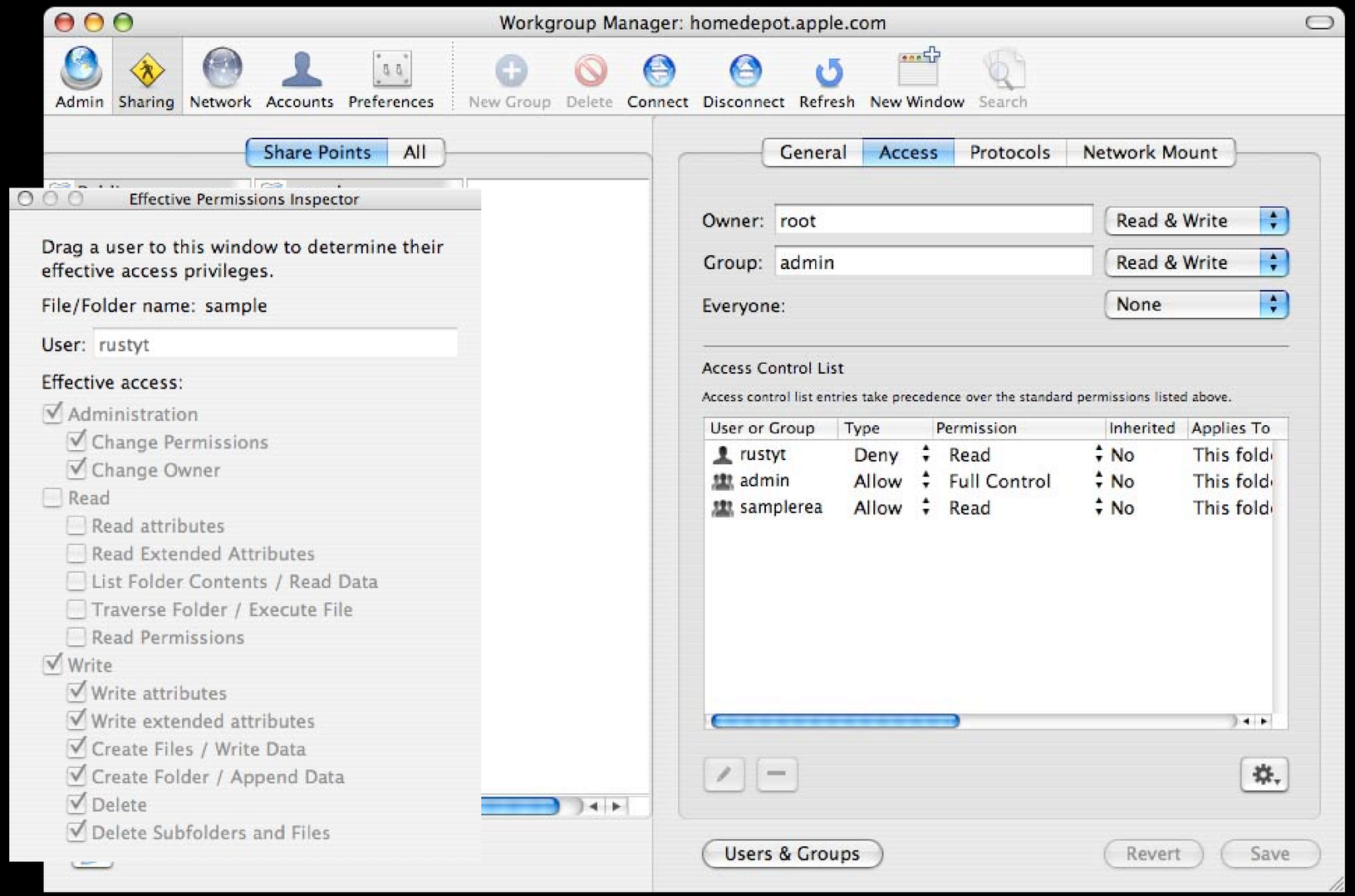
ACL Best Practices

Use the Effective Permissions Inspector



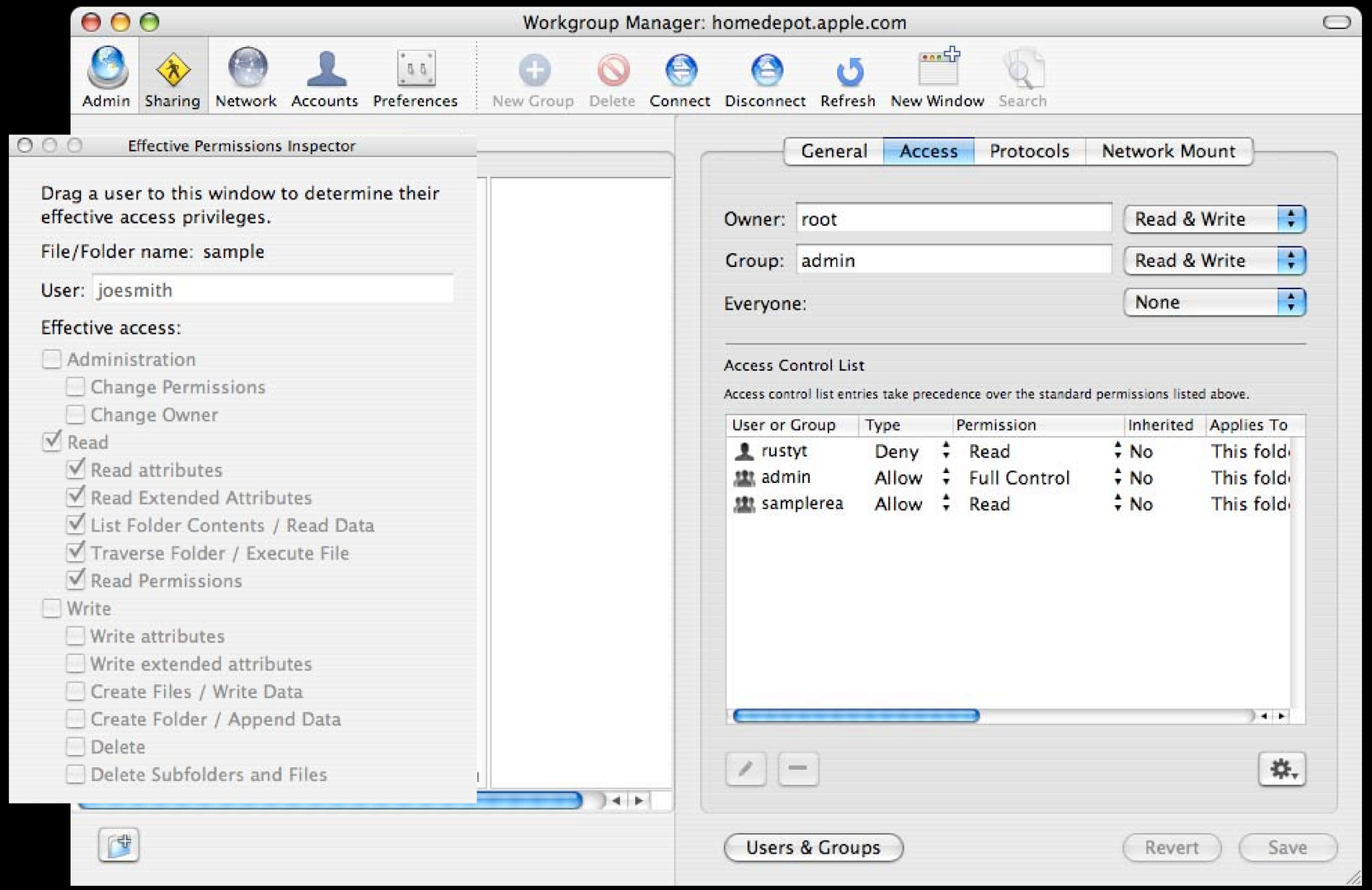
ACL Best Practices

Use the Effective Permissions Inspector



ACL Best Practices

Use the Effective Permissions Inspector



ACL Best Practices

Keep it simple

- In many cases it makes sense just to use POSIX permissions
  - /, /System, /Library
- Limit use to share points
  - Shares with a simple structure are a great starting point

ACL Best Practices

Think. Think some more. Rethink.

- Before you ever touch the keyboard
  - What do you want to accomplish?
  - Draw out the basic folder structure on paper
  - Design a Group structure
  - Decide what groups have access to which folders
  - Non-ACL clients vs. ACL clients
- Draw it all out on a white-board
- Think about it some more

Roadmap

612 Moving to Tiger Server from Windows: A Primer	Session 612 Wednesday 9:00AM
Mac OS X Server Feedback Forum	Session 614 Wednesday 10:30AM
Lab: Writing File Systems Using Supported KPIs	Performance and Tools Lab Wednesday 3:30PM Hands On

Q&A

Skip Levens

http://developer.apple.com/wwdc2005/

More Information

http://developer.apple.com/wwdc2005/

Documentation, Sample Code, and Other Resources

http://developer.apple.com/wwdc2005/

WWDC2005

Worldwide Developers Conference